BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 100 36 278.8

Anmeldetag: 26. Juli 2000

Anmelder/Inhaber: Robert Bosch GmbH, Stuttgart/DE

Bezeichnung: Verfahren zur Überwachung eines Programmablaufs mittels einer Debug Logik

IPC: G 06 F 11/00

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 23. Mai 2001
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Brand

R.38402

17.07.2000
Robert Bosch GmbH, 70469 Stuttgart

Verfahren zur Überwachung eines Programmablaufs mittels einer Debug Logik

Die vorliegende Erfindung betrifft ein Verfahren zur Überwachung des Ablaufs eines auf mindestens einem Mikroprozessor eines Mikrocontrollers ablauffähigen Programms mittels einer Debug Logik des Mikrocontrollers, wobei von der Debug Logik beim Zugriff auf einen bestimmten Adressbereich während der Programmlaufzeit eine Ausnahmebedingung (Exception), insbesondere eine Unterbrechung (Interrupt) des Programmablaufs, ausgelöst wird.

Die Erfindung betrifft außerdem ein Steuerelement, insbesondere ein Read-Only-Memory oder ein Flash-Memory, für einen Mikrocontroller insbesondere eines Steuergeräts eines Kraftfahrzeugs, wobei auf dem Steuerelement ein Programm abgespeichert ist, das auf mindestens einem Mikroprozessor des Mikrocontrollers ablauffähig und dazu geeignet ist, beim Hochfahren des Mikrocontrollers eine Debug Logik des Mikrocontrollers zu veranlassen, den Ablauf eines weiteren auf dem mindestens einen Mikroprozessor des Mikrocontrollers ablauffähigen Programms während der Programmlaufzeit zu überwachen, beim Zugriff auf einen bestimmten Adressbereich eine Ausnahmebedingung (Exception), insbesondere eine Unterbrechung (Interrupt) des Programmablaufs, auszulösen und danach eine Ausnahmebedingungs-Routine (Exception-Routine) zu

durchlaufen.

Schließlich betrifft die vorliegende Erfindung einen Mikrocontroller mit mindestens einem Mikroprozessor und einer Debug Logik, wobei auf dem mindestens einen Mikroprozessor ein Programm ablauffähig ist, die Debug Logik während der Programmlaufzeit den Ablauf des Programms überwacht und beim Zugriff auf einen bestimmten Adressbereich eine Ausnahmebedingung (Exception), insbesondere eine Unterbrechung (Interrupt) des Programmablaufs, auslöst.

Stand der Technik

Aus dem Stand der Technik sind Mikrocontroller bekannt, die mindestens einen Mikroprozessor (Rechnerkern), einen Analog/Digital (A/D)-Wandler, einen Digital/Analog (D/A)-Wandler, einen Datenbus, einen Adressbus, einen Controlbus, interne Steuerelemente (z. B. ein Read-Only-Memory oder ein Flash-Memory) und/oder weitere Bauelemente umfassen. Ein derartiger Mikrocontroller ist bspw. Teil eines Steuergerätes für ein Kraftfahrzeug. Das Steuergerät dient zur Steuerung/Regelung von technischen Vorgängen oder Prozessen in dem Kraftfahrzeug, z. B. der Brennkraftmaschine, des Getriebes, der Lenkung, des Fahrwerks, etc. In einem internen oder externen Steuerelement des Mikrocontrollers ist ein Steuerprogramm zur Ausführung der Steuerung/Regelung abgespeichert. Das Steuerprogramm ist auf mindestens einem der Mikroprozessoren ablauffähig.

Komplexere Mikrocontroller neuerer Bauart umfassen als weiteres Bauelement eine sog. Debug Logik. Die Debug Logik wird während der Entwicklung des auf dem mindestens einen Mikroprozessor des Mikrocontrollers ablauffähigen Programms

eingesetzt und dient zur Verbesserung der Sichtbarkeit der Abläufe in dem Mikrocontroller. Mit Hilfe der Debug Logik können Fehler in dem Programm erkannt, lokalisiert und aus dem Programm entfernt werden. Die Debug Logik ist an einen Datenbus, einen Adressbus und/oder einen Controlbus des Mikrocontrollers angeschlossen. Dem Adressbus kann die Debug Logik entnehmen, auf welchen ausgewählten Adressbereich zugegriffen wird, dem Datenbus, welche Daten in den ausgewählten Adressbereich geschrieben werden sollen bzw. aus dem ausgewählten Adressbereich gelesen wurden, und dem Controlbus, ob auf den ausgewählten Adressbereich schreibend oder lesend zugegriffen werden soll.

Die Debug Logik weist darüber hinaus eine Debug-Schnittstelle auf, an die ein sog. Debugger angeschlossen werden kann. Der Debugger ist üblicherweise als ein herkömmlicher Personal Computer (PC) ausgebildet, auf dem ein Debug-Programm abläuft. Mithilfe des Debuggers kann der Programmablauf auf dem Mikroprozessor gesteuert und dabei der Zustand des Programms und des Mikrocontrollers überwacht werden. Nach Abschluss der Entwicklung des Programms wird die Debug Logik nicht mehr benötigt. Da sie jedoch Teil des Mikrocontrollers ist, wird sie während des bestimmungsgemäßen Einsatzes des Mikrocontrollers während der Programmlaufzeit zwangsläufig mitgeführt, obwohl sie eigentlich keine Aufgabe hat.

Über eine event-Leitung steht die Debug Logik mit dem Mikroprozessor in Verbindung. Die Debug Logik kann in der Regel eine Ausnahmebedingung, eine sog. Exception, bspw. eine Unterbrechung, einen sog. Interrupt, auslösen. Als Interrupt wird ein Signal bezeichnet, das von einer Ein-/Ausgabeeinheit an den Mikroprozessor gesendet wird, wenn ein Fehler aufgetreten ist oder ein Eingriff erforderlich ist, um die Ein-/Ausgabe abzuschließen. Ein Interrupt bewirkt normalerweise, dass die Ausführung des

aktuellen Programms ausgesetzt und eine Interrupt-Routine ausgeführt wird.

Aus der US 5,809,293 ist es bekannt, während der Entwicklung eines auf einem Mikroprozessor eines Mikrocontrollers ablauffähigen Programms ein sog. Trace Tool einzusetzen, das die Abläufe in dem Mikrocontroller, insbesondere in dem Mikroprozessor verfolgt und aufzeichnet. Das Sichtbarmachen der Abläufe mit dem Trace Tool stößt jedoch an seine Grenzen, wenn bspw. zwischen dem Trace Tool und dem Mikroprozessor ein Zwischenspeicher (Cache) angeordnet ist, in den während der Programmlaufzeit Werte zwischengespeichert und bei Bedarf ausgelesen werden. Da das Trace Tool in einem solchen Fall nicht unmittelbar auf den Mikroprozessor, sondern nur auf die zwischengespeicherten Werte Zugriff hat, weiß es nicht, welche Befehle von dem Mikroprozessor wann ausgeführt werden. Die US 5,809,293 schlägt vor, in einem solchen Fall eine Debug Logik zur Unterstützung des Trace Tools und zur Verbesserung der Sichtbarkeit der Abläufe in dem Mikrocontroller einzusetzen.

Aus der US 5,680,620 ist ein Mikrocontroller mit einer Debug Logik bekannt, die verwendet wird, um eine Umadressierung, ein sog. Remapping, von Dateneingängen bzw. Datenausgängen, sog. Input/Output (I/O)-Ports, auszuführen. Dies kann bspw. notwendig sein, wenn ein altes Programm auf einer neuen Hardware-Umgebung abläuft, in der bestimmte I/O-Ports nicht mehr vorhanden sind. Auf die Adressen der nicht mehr vorhandenen I/O-Ports wird ein Unterbrechungspunkt, ein sog. Breakpoint, gesetzt. Die Debug Logik überwacht den Programmablauf während der Programmlaufzeit. Wenn versucht wird, auf eine solche Adresse zuzugreifen, wird eine Exception ausgelöst und eine Exception-Routine ausgeführt. Im Rahmen der Exception-Routine wird die Adresse des nicht vorhandenen I/O-Ports

auf der Andresse eines in der neuen Hardware-Umgebung vorhandenen I/O-Ports umadressiert.

Dieses aus der US 5,680,620 bekannte Verfahren kann auch angewandt werden, um ein altes Programm in einer neuen Betriebssystem-Umgebung ausführen zu können. Wenn bspw. das neue Betriebssystem anders als das alte den Zugriff auf ein I/O-Port aus mehreren Anwendungen heraus zulässt, dürfen die Zugriffe aus einer Anwendung nicht - wie unter der alten Betriebssystem-Umgebung üblich - direkt auf den I/O-Port erfolgen, sondern müssen über das Betriebssystem geleitet und von diesem koordiniert werden. In diesem Fall löst jeder Zugriff auf einen I/O-Port eine Exception aus, durch die der Zugriff zur Koordination an das Betriebssystem weitergeleitet wird.

Bei dem aus dem Stand der Technik bekannten Verfahren werden nur zulässige Zugriffe auf die I/O-Ports behandelt. Im Rahmen des bekannten Verfahrens findet keine Überprüfung der Befehle bzw. des Programms auf mögliche Fehlerzustände statt. Bei dem bekannten Verfahren handelt es sich also um eine reine Betriebssystemerweiterung.

Aus dem Stand der Technik ist es des weiteren bekannt, zur Überwachung eines Stacks eines Mikrocontrollers während des Hochfahrens des Mikrocontrollers in einen Speicherbereich am Rande des Stacks ein Muster zu schreiben und während der Programmlaufzeit von Zeit zu Zeit zu prüfen, ob dieses Muster verändert wird oder nicht. Eine Veränderung des Musters deutet auf einen Stacküberlauf oder Stackunterlauf hin. Eine derartige Überwachung eines Speicherbereichs hat jedoch den Nachteil, dass sie relativ Resourcenintensiv ist, insbesondere kostet sie Rechenleistung des Mikroprozessors und Programmspeicher. Außerdem ist es bei modernen Mikrocontrollern zur Überwachung eines Stacks nicht mehr ausreichend, eine Veränderung eines Musters in

einem begrenzten Speicherbereich am Rande des Stacks zu überwachen, da beim Aufbau eines Stackframes auch ein Sprung über mehrere Adressen stattfinden kann, ohne dass die Adressen in dem Zwischenraum verändert werden. Aus diesem Grund muss der zu überwachende Speicherbereich am Rande des Stacks immer größer gewählt werden, wodurch die Resourcen des Mikrocontrollers stark belastet werden und die Rechenzeit ansteigt. Außerdem erkennt das bekannte Verfahren einen Stacküberlauf bzw. einen Stackunterlauf erst, wenn bereits auf Speicherbereiche außerhalb des vorgesehenen Stackbereichs zugegriffen worden ist. Dadurch besteht die Gefahr, dass das auf dem Mikroprozessor ablauffähige Programm in einen undefinierten Zustand gerät noch bevor das Überschreiten oder Unterschreiten des vorgesehenen Stackbereichs detektiert wird und entsprechende Gegenmassnahmen eingeleitet werden können.

Die Aufgabe der vorliegenden Erfindung besteht darin, eine besonders zuverlässige aber dennoch möglichst resourcenschonende Überwachung des Ablaufs eines auf einem Mikroprozessor ablauffähigen Programms auf Fehlerzustände zu schaffen.

Zur Lösung der Aufgabe schlägt die Erfindung ausgehend von dem Verfahren der eingangs genannten Art vor, dass die Debug Logik von dem Mikroprozessor konfiguriert wird und von der Debug Logik nach dem Auslösen einer Ausnahmebedingung während der Programmlaufzeit eine Ausnahmebedingungs-Routine (Exception-Routine) durchlaufen wird.

Vorteile der Erfindung

Gemäß der vorliegenden Erfindung wird eine Debug Logik, die während der Programmlaufzeit in der Regel ohne eine Aufgabe

ist, zur Überwachung des Programms auf Fehlerzustände herangezogen. Die von der Debug Logik zu detektierenden Fehlerzustände können unterschiedlicher Art sein. Insbesondere ist an die Überwachung von bestimmten Speicherbereichen gedacht, um einen unerlaubten Zugriff auf diese Speicherbereiche zu verhindern. Es ist denkbar, dass das Programm aufgrund von Programmierungs- oder Hardwarefehlern während der Programmlaufzeit auf physisch nicht vorhandene oder auf Speicherbereiche zugreifen möchte, die außerhalb eines vorgesehenen Speicherbereichs liegen. Ein solcher unerlaubter Zugriff kann das Programm in einen undefinierten Zustand und zu unvorhersehbaren Aktionen des Steuergeräts führen.

Der Einsatz der Debug Logik zur Fehleranalyse eines Programms während der Programmlaufzeit hat den Vorteil, dass die Debug Logik eine zuverlässige Überwachung des Programmablaufs ermöglicht. Außerdem handelt es sich um eine besonders resourcenschonende Überwachung des Programmablaufs, da die Debug Logik weder Rechenleistung des Mikroprozessors noch Programmspeicher beansprucht.

Mit dem erfindungsgemäßen Verfahren kann die Sicherheit eines Mikrocontrollers auf verschiedene Weise erhöht werden. Zunächst ist eine höhere Rechenbelastung des Mikroprozessors möglich. Außerdem ist eine zuverlässige Funktion der Fehleranalyse unabhängig von der Rechenbelastung des Mikroprozessors gegeben. Schließlich ist die Implementierung der Fehleranalyse unabhängig von der verwendeten Speicherrealisierung. Bei einem Stack ist es bspw. unerheblich, wie ein Stackframe angelegt wird. Selbst ein Sprung über mehrere Speicheradressen beim Aufbau des Stackframes führt nicht zum Verlust der Überwachungssicherheit.

Gemäß einer vorteilhaften Weiterbildung der vorliegenden

Erfindung wird vorgeschlagen, dass die Debug Logik während des Hochfahrens des Mikrocontrollers konfiguriert wird.

Gemäß einer bevorzugten Ausführungsform der vorliegenden Erfindung wird vorgeschlagen, dass der Mikrocontroller während des Durchlaufs der Exception-Routine rückgesetzt und erneut hochgefahren und das überwachte Programm initialisiert wird.

Um zu verhindern, dass das Programm aufgrund des aufgetretenen Fehlerzustandes in einen undefinierten Zustand gerät und das Steuergerät unvorhersehbare Aktionen ausführt, wird der Mikrocontroller nach dem Auslösen der Ausnahmebedingung sicherheitshalber rückgesetzt und neu hochgefahren und das Programm initialisiert. Nach dem Hochfahren des Mikrocontrollers und dem Initialisieren des Programms befindet sich dieses in einem definierten Ausgangszustand.

Gemäß einer anderen bevorzugten Ausführungsform der Erfindung wird vorgeschlagen, dass vor dem Rücksetzen und dem erneuten Hochfahren des Mikrocontrollers und vor der Initialisierung des Programms zumindest die Art des Fehlerszustandes in einem Fehlerspeicher abgelegt wird. Der Inhalt des Fehlerspeichers kann von Zeit zu Zeit, bspw. bei einem Werkstattaufenthalt des Kraftfahrzeugs, ausgelesen und analysiert werden. Aus dem Fehlerspeicher lassen sich wichtige Informationen über die Zuverlässigkeit des Programms und damit auch über die Sicherheit des Steuergeräts entnehmen. Vorteilhafterweise wird vor dem Rücksetzen und dem erneuten Hochfahren des Mikrocontrollers und vor der Initialisierung des Programms auch die Speicheradresse, auf die vor Auftreten des Fehlerzustandes zugegriffen wurde, in dem Fehlerspeicher abgelegt.

Gemäß einer vorteilhaften Weiterbildung der vorliegenden

Erfindung wird vorgeschlagen, dass von der Debug Logik überwacht wird, ob das Programm während der Programmlaufzeit auf einen vorgebbaren Adressbereich eines Speichers zugreift. Die Debug Logik steht mit einem Adressbus, einem Datenbus und/oder einem Controlbus in Verbindung. Dem Adressbus kann die Debug Logik entnehmen, auf welchen Adressbereich zugegriffen wird, dem Datenbus, welche Daten in einen bestimmten Adressbereich geschrieben bzw. aus einem bestimmten Adressbereich gelesen wurden, und dem Controlbus, ob auf einen bestimmten Adressbereich schreibend oder lesend zugegriffen wird. Der Adressbereich kann bspw. einem bestimmten Speicherbereich zugeordnet sein.

Gemäß einer bevorzugten Ausführungsform der Erfindung wird vorgeschlagen, dass von der Debug Logik überwacht wird, ob das Programm während der Programmlaufzeit auf einen Adressbereich eines Stacks des Mikrocontrollers jenseits einer vorgebbaren maximalen Stackgröße zugreift. Jenseits der maximalen Stackgröße wird ein illegaler Speicherbereich, eine sog. Breakregion, definiert, auf den ein Unterbrechungspunkt, ein sog. Breakpoint gesetzt wird. Wenn ein Stackpointer während der Programmlaufzeit auf diesen illegalen Speicherbereich zeigt, falls also versucht wird, auf den illegalen Speicherbereich zuzugreifen, wird eine Ausnahmebedingung, eine sog. Exception, ausgelöst.

Der Stack ist der einzige Speicherbereich, dessen Größe nicht genau festgelegt ist und sich während der Programmlaufzeit ändern kann. Zur Einsparung von Speicherplatz wird der maximale Stackbereich so klein wie möglich gewählt. Andererseits muss er jedoch ausreichend groß gewählt werden, um auch im worst case sicherzustellen, dass der Stackpointer stets auf eine Adresse innerhalb des maximalen Speicherbereichs zeigt. In bestimmten Betriebsituationen des Mikrocontrollers kann es dazu

kommen, dass so weit gefüllt ist, dass der Stackpointer auf den illegalen Speicherbereich zeigt. Diese Situationen können mit dem erfindungsgemäßen Verfahren besonders zuverlässig und gleichzeitig resourcenschonend detektiert werden.

Gemäß einer alternativen Weiterbildung der vorliegenden Erfindung wird vorgeschlagen, dass von der Debug Logik überwacht wird, ob während der Programmlaufzeit versucht wird, eine aus einem Flash-Speicher des Mikrocontrollers in einen Random-Access-Speicher des Mikrocontrollers ausgelagerte Code-Sequenz des Programms in dem Flash-Speicher zur Ausführung zu bringen. Ein Update eines Flash-Speichers während der Programmlaufzeit wird in der Regel über einen Contoller Area Network (CAN)-Bus oder eine Diagnoseschnittstelle ausgeführt, da diese bei einem geschlossenen Steuergerät üblicherweise die einzigen von außen frei zugänglichen Schnittstellen sind. Diese Schnittstellen werden von bestimmten Codesequenzen eines Programms gesteuert. Der Update des Flash-Speichers kann deshalb nur während der Ausführung dieser Codesequenzen ausgeführt werden. Aus diesem Grund ist es notwendig während des Updates des Flash-Speichers die zur Steuerung der Schnittstellen erforderlichen Codesequenzen des Programms aus dem Flash-Speicher in ein Random-Access-Memory zu übertragen, wo sie ausgeführt werden müssen. Wenn nun während der Programmlaufzeit versucht wird, auf den Flash-Speicher zur Ausführung der Codesequenzen zuzugreifen, deutet dies darauf hin, dass sich ein Programmzeiger versprungen hat. Es besteht die Gefahr, dass das Programm in einen undefinierten Zustand gerät. Aus diesem Grund wird von der Debug Logik eine Exception ausgelöst, falls während der Programmlaufzeit versucht wird, auf den Flash-Speicher zur Ausführung der Codesequenzen zuzugreifen.

Im Rahmen der vorliegenden Erfindung kann von der Debug Logik auch die Auslastung des Mikroprozessors des Mikrocontrollers während der Programmlaufzeit überwacht werden. Dazu wird von der Debug Logik Anfang und Ende einer Idle-Task erfasst und mithilfe eines weiteren Mikroprozessors protokolliert. Der Mikroprozessor führt dann eine Idle-Task aus, wenn er sonst keine Tasks ausführen muss, wenn er also nicht ausgelastet ist. Die Häufigkeit der Ausführung der Idle-Task ist also umgekehrt proprotional der Auslastung des Mikrocontrollers. Die Speicheradressen der Idle-Task sind bekannt. Die Debug Logik kann einen Zugriff auf diese Speicheradressen überwachen und zu Beginn und Ende der Idle-Task eine Exception auslösen. Der weitere Mikroprozessor kann die Event-Leitung überwachen, über die die Debug Logik mit dem Mikroprozessor in Verbindung steht. Der weitere Mikroprozessor detektiert das Auslösen einer Exception und kann dadurch Anfang und Ende der Idle-Task protokollieren. Bei einem Mikrocontroller mit mehreren Mikroprozessoren kann ein erster Prozessor des Mikrocontrollers zur Ausführung des Programms und ein anderer Prozessor des Mikrocontrollers zur Überwachung der Auslastung des ersten Prozessors herangezogen werden.

Von besonderer Bedeutung ist die Realisierung des erfindungsgemäßen Verfahrens in Form eines Steuerelements, das für einen Mikrocontroller insbesondere eines Steuergeräts eines Kraftfahrzeugs vorgesehen ist. Dabei ist auf dem Steuerelement ein Programm abgespeichert, das auf mindestens einem Mikroprozessor eines Mikrocontrollers ablauffähig und zur Ausführung eines erfindungsgemäßen Verfahrens geeignet ist. Insbesondere ist das Steuerelement dazu geeignet, beim Hochfahren des Mikrocontrollers eine Debug Logik des Mikrocontrollers zu veranlassen, den Ablauf eines weiteren auf dem mindestens einen Mikroprozessor des Mikrocontrollers ablauffähigen Programms während der

Programmlaufzeit zu überwachen. Durch das Programm wird ein Breakpoint auf einen bestimmten Adressbereich gelegt, so dass bei einem Zugriff auf diesen Adressbereich eine Ausnahmebedingung (Exception), insbesondere eine Unterbrechung (Interrupt) des Programmablaufs, ausgelöst wird. Das Programm umfasst auch eine besondere Ausnahmebedingungs-Routine (Exception-Routine), die im Anschluss an die Ausnahmebedingung durchlaufen wird. Im Rahmen der Ausnahmebedingungs-Routine wird der Mikrocontroller rückgesetzt und erneut hochgefahren und das überwachte Programm initialisiert. Das weitere, zu überwachende Programm ist bspw. als ein Steuerprogramm des Steuergeräts ausgebildet. Als Steuerelement kann insbesondere ein elektrisches Speichermedium zur Anwendung kommen, bspw. ein Read-Only-Memory oder ein Flash-Memory.

Als eine weitere Lösung der Aufgabe der vorliegenden Erfindung wird ausgehend von dem Mikrocontroller der eingangs genannten Art vorgeschlagen, dass der Mikroprozessor die Debug Logik konfiguriert und der Mikrocontroller Mittel zum Durchlaufen einer Ausnahmebedingungs-Routine (Exception-Routine) nach dem Auslösen einer Ausnahmebedingung während der Programmlaufzeit aufweist.

Gemäß einer vorteilhaften Weiterbildung der Erfindung wird vorgeschlagen, dass der Mikrocontroller weitere Mittel zur Ausführung eines Verfahrens nach einem der Ansprüche 1 bis 8 aufweist.

Zeichnungen

Weitere Merkmale, Anwendungsmöglichkeiten und Vorteile der Erfindung ergeben sich aus der nachfolgenden Beschreibung von Ausführungsbeispielen der Erfindung, die in der

Zeichnung dargestellt sind. Dabei bilden alle beschriebenen oder dargestellten Merkmale für sich oder in beliebiger Kombination den Gegenstand der Erfindung, unabhängig von ihrer Zusammenfassung in den Patentansprüchen oder deren Rückbeziehung sowie unabhängig von ihrer Formulierung bzw. Darstellung in der Beschreibung bzw. in der Zeichnung. Es zeigen:

Fig. 1 einen erfindungsgemäßen Mikrocontroller gemäß einer ersten bevorzugten Ausführungsform der Erfindung;

Fig. 2 einen erfindungsgemäßen Mikrocontroller gemäß einer zweiten bevorzugten Ausführungsform der Erfindung; und

Fig. 3 ein Ablaufdiagramm des erfindungsgemäßen Verfahrens.

Beschreibung der Ausführungsbeispiele

In Fig. 1 ist ein Mikrocontroller gemäß der vorliegenden Erfindung in seiner Gesamtheit mit dem Bezugszeichen 1 bezeichnet. Der Mikrocontroller 1 ist bspw. Teil eines Steuergerätes 2 für ein Kraftfahrzeug. Das Steuergerät 2 dient zur Steuerung/Regelung von technischen Vorgängen oder Prozessen in dem Kraftfahrzeug, z. B. der Brennkraftmaschine, des Getriebes, der Lenkung, des Fahrwerks, etc. Der Mikrocontroller 1 weist einen Mikroprozessor 3, auf dem ein Steuerprogramm zur Ausführung der Steuerung/Regelung ablauffähig ist. Das Steuerprogramm ist in einem internen Steuerelement 4 und/oder in einem externen Steuerelement 5 abgespeichert. Die Steuerelemente 4, 5 sind vorzugsweise als Speicherelemente ausgebildet. Das interne Steuerelement 4 ist als ein Flash-Speicher und

das externe Steuerelement 5 als ein Random-Access-Memory ausgebildet. Das externe Steuerelement 5 ist über ein Bus-Interface 13 an den Datenbus D, den Adressbus A und den Controlbus C angeschlossen. Das Bus-Interface 13 wandelt die Bussignale in den Standard des externen Steuerelements 5 um.

Der Mikrocontroller 1 weist des weiteren eine Debug Logik 6 auf. Die Debug Logik 6 wird nach dem Stand der Technik während der Entwicklung des auf dem Mikroprozessor 3 des Mikrocontrollers 1 ablauffähigen Steuerprogramms eingesetzt und dient zur Verbesserung der Sichtbarkeit der Abläufe in dem Mikrocontroller 1. Mit Hilfe der Debug Logik 6 können Fehler in dem Steuerprogramm erkannt, lokalisiert und aus dem Programm entfernt werden. Der Mikroprozessor 3, das interne Steuerelement 4, das externe Steuerelement 5 und die Debug Logik 6 stehen über einen Datenbus D, einen Adressbus A und/oder einen Controlbus C des Mikrocontrollers 1 untereinander in Verbindung.

Die Debug Logik 6 weist eine von außerhalb des Steuergeräts 2 zugänglich an das Gehäuse des Steuergeräts 2 geführte Debug-Schnittstelle 7 auf, an die ein sog. Debugger 8 angeschlossen werden kann. Der Debugger 8 ist üblicherweise als ein herkömmlicher Personal Computer (PC) ausgebildet, auf dem ein Debug-Programm abläuft. Mithilfe des Debuggers 8 kann der Ablauf des Programms auf dem Mikroprozessor 3 gesteuert und dabei der Zustand des Programms und des Mikrocontrollers 1 überwacht werden.

Über eine event-Leitung 9 steht die Debug Logik 6 mit dem Mikroprozessor 3 in Verbindung. Die Debug Logik 6 kann eine Ausnahmebedingung, eine sog. Exception, bspw. eine Unterbrechung, ein sog. Interrupt, erzeugen und die Exception über die event-Leitung 9 an den Mikroprozessor 3 leiten. Beim Auftreten einer bestimmten Exception wird aus

einer Exception-Tabelle eine entsprechende Exception-Routine ausgewählt. Die Ausführung des aktuellen Programms wird ausgesetzt und die Exception-Routine durchlaufen. Eine Exception-Routine kann verschiedene Aktionen veranlassen, die von einem Rücksetzen und erneutem Hochfahren des Mikrocontrollers 1 und einer Initialisierung des Programms bis zu einem Bedienen bestimmter Bauelemente (z. B. von Ein-/Ausgabeeinheiten) des Mikrocontrollers 1 reichen.

Erfindungsgemäß wird die Debug Logik 6 während der Laufzeit des auf dem Mikroprozessor 3 ausführbaren Programms zur Überwachung des Programms auf das Auftreten eines vorgebbaren Fehlerzustandes eingesetzt. Nach dem Auftreten des vorgegebenen Fehlerzustandes wird über die event-Leitung 9 eine Ausnahmebedingung ausgelöst und eine dem Fehlerzustand entsprechende Exception-Routine durchlaufen. Im Rahmen der Exception-Routine wird zunächst die Art des Fehlerszustandes und die Speicheradresse, auf die vor dem Auftreten des Fehlerzustandes zugegriffen wurde, in einem Fehlerspeicher (nicht dargestellt) abgelegt. Danach wird der Mikrocontroller 1 rückgesetzt und erneut hochgefahren und das überwachte Steuerprogramm initialisiert. Der Inhalt des Fehlerspeichers kann von Zeit zu Zeit, bspw. bei einem Werkstattaufenthalt des Kraftfahrzeugs, ausgelesen und analysiert werden.

Die Fehleranalyse gemäß dem erfindungsgemäßen Verfahren wird vorzugsweise zum Überwachen eines vorgebbaren Adressbereichs eines Speichers eingesetzt. Insbesondere wird das Steuerprogramm dahingehend überwacht, ob während der Programmlaufzeit auf einen illegalen Adressbereich 10 eines Stacks 11 des Mikrocontrollers 1 jenseits einer vorgebbaren maximalen Stackgröße von 4 kB zugreift. Ein Stapelspeicher, ein sog. Stack 11, ist ein Bereich eines Random-Access-Memory (RAM) 12 des Mikrocontrollers 1. Bei dem Stack 11 handelt sich um eine Datenstruktur, bei der

Datenelemente am Ende einer sequentiellen Liste hinzugefügt (in den Stack 11 eingegeben) und von demselben Ende wieder abgerufen (aus dem Stack 11 entnommen) werden. Diese Zugriffsart entspricht dem Last-In/First-Out (LIFO)-Verfahren. In dem vorliegenden Ausführungsbeispiel hat das RAM 12 eine Größe von 26 kB, die maximale Stackgröße beträgt 4 kB.

Auf den illegalen Speicherbereich 10 wird ein Unterbrechungspunkt, ein sog. Breakpoint gesetzt. Der Speicherbereich 10 wird auch als Breakregion bezeichnet. Sobald ein Stackpointer auf den illegalen Speicherbereich 10 zeigt, löst die Debug Logik 6 über die event-Leitung 9 eine Exception aus. Der Stackpointer ist bspw. als ein vordefiniertes Register ausgebildet, dessen Dateninhalt dem Adressbereich des Stacks 11 entspricht, auf den der Stackpointer zeigt. Wenn auf dem Adressbus A bspw. ein Wert liegt, der innerhalb des illegalen Speicherbereichs 10 liegt, wird dies von der Debug Logik 6 registriert. Wird dann zusätzlich noch an den Controlbus C ein Befehl zum Schreiben oder Lesen angelegt, wenn also versucht wird, auf den illegalen Speicherberich 10 zuzugreifen, löst die Debug Logik 6 die Exception aus.

In Fig. 2 ist eine alternativ Ausführungsform der vorliegenden Erfindung dargestellt. Ein Update des als Flash-Speicher ausgebildeten internen Steuerelements 4 während der Programmlaufzeit wird in der Regel über einen Contoller Area Network (CAN)-Bus 14 oder eine Diagnoseschnittstelle 15 ausgeführt, da diese bei einem geschlossenen Steuergerät 2 üblicherweise die einzigen von außen frei zugänglichen Schnittstellen sind. Diese Schnittstellen 14, 15 werden von bestimmten Codesequenzen eines auf dem Mikroprozessor 3 ablauffähigen Programms gesteuert. Der Update des internen Steuerelements 4 kann deshalb nur während der Ausführung dieser Codesequenzen

durchgeführt werden. Aus diesem Grund ist es notwendig während des Updates des internen Steuerelements 4 die zur Steuerung der Schnittstellen 14, 15 erforderlichen Codesequenzen des Programms aus dem internen Steuerelement 4 in den bspw. als Random-Access-Memory ausgebildeten externen Speicher 5 zu übertragen, wo sie während des Updates ausgeführt werden müssen.

Es ist denkbar, dass das Steuerprogramm aufgrund von Programmierungs- oder Hardwarefehlern während der Programmlaufzeit fälschlicherweise auf das internen Steuerelement 4 zur Ausführung der ausgelagerten Codesequenzen zugreifen möchte. Dies deutet darauf hin, dass sich ein Programmzeiger versprungen hat. Es besteht die Gefahr, dass das Programm in einen undefinierten Zustand gerät. Um dies zu vermeiden wird gemäß der zweiten bevorzugten Ausführungsform der Erfindung von der Debug Logik 6 überwacht, ob während des Ablaufs des Steuerprogramms versucht wird, die ausgelagerte Code-Sequenz des Steuerprogramms in dem internen Steuerelement 4 zur Ausführung zu bringen. Die Debug Logik 6 löst eine Exception aus, falls während der Programmlaufzeit versucht wird, auf den internen Speicher 4 zur Ausführung der Codesequenzen zuzugreifen.

In Fig. 3 ist ein Ablaufdiagramm des erfindungsgemäßen Verfahrens für das Ausführungsbeispiel aus Fig. 1 dargestellt. Das Verfahen beginnt in Funktionsblock 20. Anschließend wird in einem Funktionsblock 21 der Mikrocontroller 1 hochgefahren. Während des Hochfahrens des Mikrocontrollers 1 wird in Funktionsblock 22 auf dem Mikroprozessor 3 ein auf einem Steuerelement, z. B. dem internen Steuerelement 4, abgespeichertes Programm ausgeführt. Das Programm ist dazu geeignet, beim Hochfahren des Mikrocontrollers 1 die Debug Logik 6 zu veranlassen, den Ablauf des weiteren auf dem Mikroprozessor 3 des

Mikrocontrollers 1 ablauffähigen Stuerprogramms während der Programmlaufzeit zu überwachen. Durch das Programm wird während des Hochfahrens ein Breakpoint auf den illegalen Adressbereich 10 gelegt, so dass bei einem Zugriff auf diesen Adressbereich 10 die Exception ausgelöst wird. Das Programm umfasst auch eine bestimmte Exception-Routine, die nach dem Auslösen der Exception durchlaufen wird.

In einem nachfolgenden Funktionsblock 23 wird das zu überwachende Steuerprogramm initialisiert. Anschließend wird das zu überwachende Steuerprogramm in einem Funktionsblock 24 ausgeführt, d. h. das Steuergerät 2 erfüllt seine bestimmungsgemäße Steuerungs-/ Regelungsaufgabe. Parallel dazu überwacht die Debug Logik 6 in einem Funktionsblock 25 den Ablauf des Steuerprogramms und erfüllt ihre Fehleranalyseaufgabe.

In einem Abfrageblock 26 wird überprüft, ob das Steuerprogramm während der Programmlaufzeit auf den illegalen Adressbereich 10 zugreift. Falls nein, wird wieder zu den Funktionsblöcken 24 und 25 verzweigt. Falls ja, wird eine Exception ausgelöst (Funktionsblock 27), der Ablauf des Steuerprogramms wird unterbrochen und die Exception-Routine wird durchlaufen. Im Rahmen der Exception-Routine wird zunächst in einem Funktionsblock 28 die Art des Fehlerszustandes und die Speicheradresse, auf die vor Auftreten des Fehlerzustandes zugegriffen wurde, in einem Fehlerspeicher abgelegt wird. Danach wird in einem Funktionsblock 29 der Mikrocontroller 1 rückgesetzt und erneut hochgefahren. Schließlich wird das überwachte Steuerprogramm in einem Funktionsblock 30 initialisiert und dann zu dem Funktionsblock 24 zur weiteren Ausführung des Steuerprogramms und zu Funktionsblock 25 zur parallelen Fehleranalyse verzweigt.

17.07.2000
Robert Bosch GmbH, 70469 Stuttgart

Ansprüche

1. Verfahren zur Überwachung des Ablaufs eines auf mindestens einem Mikroprozessor (3) eines Mikrocontrollers (1) ablauffähigen Programms mittels einer Debug Logik (6) des Mikrocontrollers (1), wobei von der Debug Logik (6) beim Zugriff auf einen bestimmten Adressbereich (10) während der Programmlaufzeit eine Ausnahmebedingung (Exception), insbesondere eine Unterbrechung (Interrupt) des Programmablaufs, ausgelöst wird, **dadurch gekennzeichnet**, dass die Debug Logik (6) von dem Mikroprozessor (3) konfiguriert wird und von der Debug Logik (6) nach dem Auslösen einer Ausnahmebedingung während der Programmlaufzeit eine Ausnahmebedingungs-Routine (Exception-Routine) durchlaufen wird.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass die Debug Logik (6) während des Hochfahrens des Mikrocontrollers (1) konfiguriert wird.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet, dass der Mikrocontroller (1) während des Durchlaufs der Exception-Routine rückgesetzt und erneut hochgefahren und das überwachte Programm initialisiert wird.

4. Verfahren nach Anspruch 3, dadurch gekennzeichnet, dass vor dem Rücksetzen und dem erneuten Hochfahren des Mikrocontrollers (1) und vor der Initialisierung des

Programms zumindest die Art des Fehlerszustandes in einem Fehlerspeicher abgelegt wird.

5. Verfahren nach Anspruch 4, dadurch gekennzeichnet, dass vor dem Rücksetzen und dem erneuten Hochfahren des Mikrocontrollers (1) und vor der Initialisierung des Programms die Speicheradresse, auf die vor Auftreten des Fehlerzustandes zugegriffen wurde, in dem Fehlerspeicher abgelegt wird.

6. Verfahren nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, dass von der Debug Logik (6) überwacht wird, ob das Programm während der Programmlaufzeit auf einen vorgebbaren Adressbereich (10) eines Speichers (12) zugreift.

7. Verfahren nach Anspruch 6, dadurch gekennzeichnet, dass von der Debug Logik (6) überwacht wird, ob das Programm während der Programmlaufzeit auf einen Adressbereich (10) eines Stacks (12) des Mikrocontrollers (1) jenseits einer vorgebbaren maximalen Stackgröße (11) zugreift.

8. Verfahren nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, dass von der Debug Logik (6) überwacht wird, ob während der Programmlaufzeit versucht wird, eine aus einem Flash-Speicher (4) des Mikrocontrollers (1) in einen Random-Access-Speicher (5) des Mikrocontrollers (1) ausgelagerte Code-Sequenz des Programms in dem Flash-Speicher (4) zur Ausführung zu bringen.

9. Steuerelement (4), insbesondere Read-Only-Memory oder Flash-Memory, für einen Mikrocontroller (1) insbesondere eines Steuergeräts (2) eines Kraftfahrzeugs, wobei auf dem Steuerelement (4) ein Programm abgespeichert ist, das auf mindestens einem Mikroprozessor (3) des Mikrocontrollers

(1) ablauffähig und zur Ausführung eines Verfahrens nach einem der Ansprüche 1 bis 8 geeignet ist.

10. Mikrocontroller (1) mit mindestens einem Mikroprozessor (3) und einer Debug Logik (6), wobei auf dem mindestens einen Mikroprozessor (3) ein Programm ablauffähig ist, die Debug Logik (6) während der Programmlaufzeit den Ablauf des Programms überwacht und beim Zugriff auf einen bestimmten Adressbereich (10) eine Ausnahmebedingung (Exception), insbesondere eine Unterbrechung (Interrupt) des Programmablaufs, auslöst, **dadurch gekennzeichnet**, dass der Mikroprozessor (3) die Debug Logik (6) konfiguriert und der Mikrocontroller (1) Mittel zum Durchlaufen einer Ausnahmebedingungs-Routine (Exception-Routine) nach dem Auslösen einer Ausnahmebedingung während der Programmlaufzeit aufweist.

11. Mikrocontroller (1) nach Anspruch 10, dadurch gekennzeichnet, dass der Mikrocontroller (1) weitere Mittel zur Ausführung eines Verfahrens nach einem der Ansprüche 1 bis 8 aufweist.

R.38402

17.07.2000
Robert Bosch GmbH, 70469 Stuttgart

Verfahren zur Überwachung eines Programmablaufs mittels einer Debug Logik

Zusammenfassung

Die Erfindung betrifft ein Verfahren zur Überwachung des Ablaufs eines auf mindestens einem Mikroprozessor (3) eines Mikrocontrollers (1) ablauffähigen Programms mittels einer Debug Logik (6) des Mikrocontrollers (1), wobei von der Debug Logik (6) beim Zugriff auf einen bestimmten Adressbereich (10) während der Programmlaufzeit eine Ausnahmebedingung (Exception), insbesondere eine Unterbrechung (Interrupt) des Programmablaufs, ausgelöst wird. Um eine besonders zuverlässige aber dennoch möglichst resourcenschonende Überwachung des Ablaufs eines auf einem Mikroprozessor (3) ablauffähigen Programms auf [illegible]hlerzustände zu schaffen, wird vorgeschlagen, dass die Debug Logik (6) von dem Mikroprozessor (3) konfiguriert wird und von der Debug Logik (6) nach dem Auslösen einer Ausnahmebedingung während der Programmlaufzeit eine Ausnahmebedingungs-Routine (Exception-Routine) durchlaufen wird. Vorteilhafterweise wird die Debug Logik (6) während des Hochfahrens des Mikrocontrollers (1) konfiguriert. Während des Durchlaufs der Exception-Routine wird vorzugsweise der Mikrocontroller (1) rückgesetzt und erneut hochgefahren und das überwachte Programm initialisiert. (Figur 1)

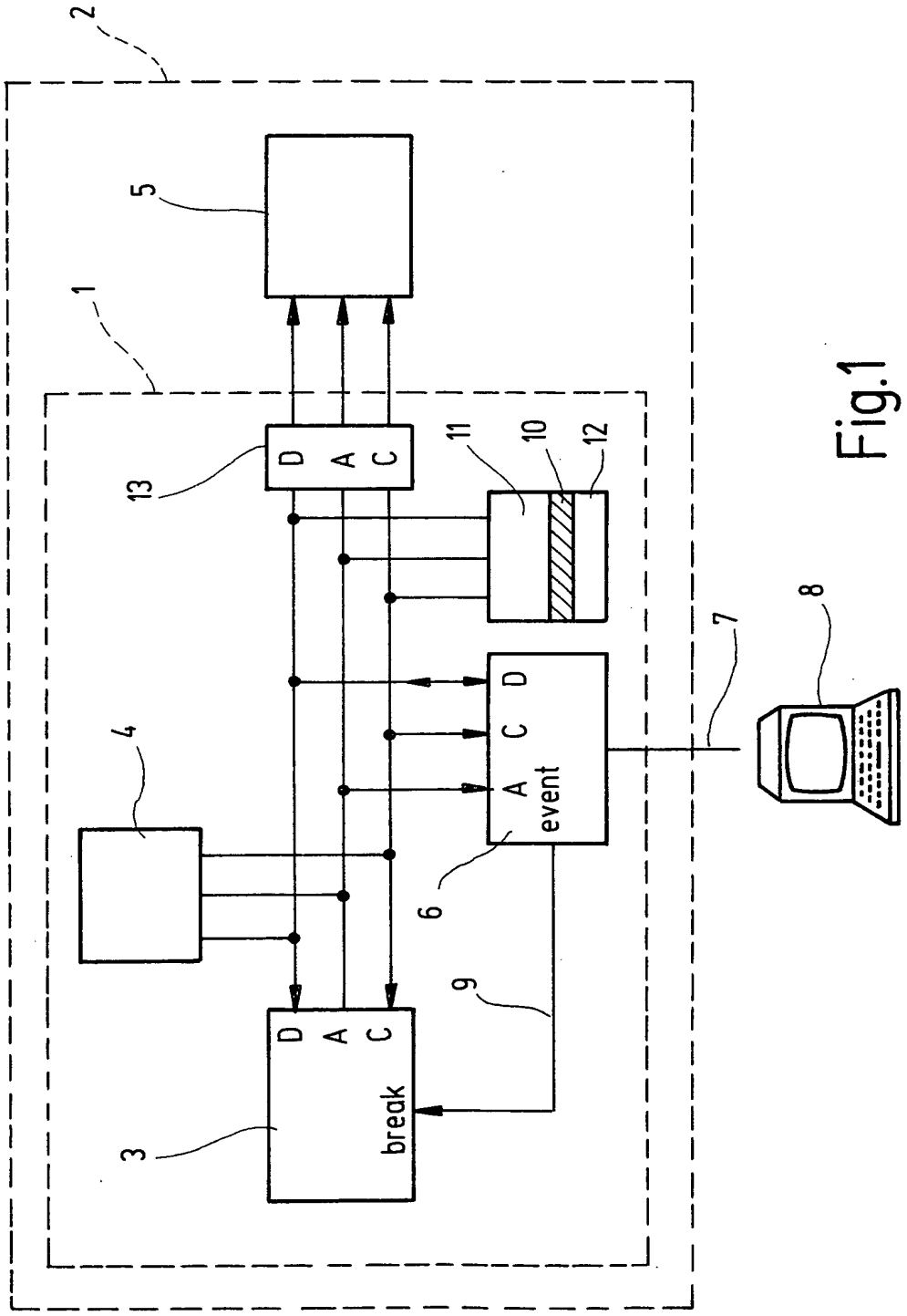

Fig.1